MODULAR, ROBUST, COMPONENT USER INTERFACE FRAMEWORK

FIELD OF THE INVENTION

[01] Aspects of the present invention relate to a modular user interface framework. More specifically, aspects of the present invention provide a componentized user interface that includes a framework and plug-ins that provide users a method to customize user interfaces.

BACKGROUND

[02] User interfaces such as graphical user interfaces are used by computer applications to accommodate the interaction between computer applications and users of computer applications. The goal of a user interface is to communicate information from the computer application to the user and from the user to the computer application. Most computer applications are packaged with user interfaces that were specifically designed for the particular computer application or hosting environment. These specifically designed user interfaces may only allow a user to make minor modifications to the user interface. Currently designed user interfaces have several disadvantages. One disadvantage is that current user interface designs are not modular or componentized and therefore specific pieces of these user interfaces cannot be used in a variety of different computer applications or hosting environments. This drawback may prevent a user from obtaining the functionality of specific user interface elements in various different computer applications or hosting environments. In addition, a user may receive additional functionality that is not required and therefore not cost effective.

[03] For example, a tool bar is a standard graphical user interface element that is employed in most current software applications. Generally, a toolbar appears to the user as a single row or column of icons or text on a computer monitor. The icons or text represent

various program functions or commands that the computer application understands. The user selects the function or command represented by the icon or text and the computer application or hosting environment carries out the program function or command. However, this same toolbar that contains various function or commands may not be supported in other computer applications or hosting environments as the other computer applications or hosting environments may have been written using different programming languages from that of the program functions or commands of the toolbar. In addition, the particular user interface may not allow for the addition of such additional toolbars.

[04] Therefore, there is a need in the art for a system and method that allows a user to utilize user interface elements in different computer applications or hosting environment without having to rewrite the underlying code.

BRIEF SUMMARY

[05] Aspects of the present invention address one or more of the issues mentioned above, thereby providing a componentized user interface that may be customized to fit a user's specific needs. A framework provides a first set of interface elements for the componentized user interface. The interface elements provided by the framework are common interface elements to a plurality of plug-ins. A second and third set of interface elements are provided by a first plug-in and a second plug-in, respectively. A shell that is linked to the framework hosts the first and the second plug-ins. A shell adapter provides an interface between the shell and the first plug-in and between the shell and the second plug-in in order to utilize both the second and third set of interface elements.

BRIEF DESCRIPTION OF THE DRAWINGS

[06] Aspects of the present invention are described with respect to the accompanying figures, in which like reference numerals identify like elements, and in which:

[07] Figure 1 illustrates a functional block diagram of a conventional general-purpose computer system;

[08] Figure 2 illustrates a system providing a componentized user interface in accordance with an embodiment of the invention;

[09] Figure 3 illustrates a screen shot illustrating a user interface shell of a componentized user interface in accordance with an embodiment of the invention;

[10] Figure 4 illustrates a system providing a componentized user interface for use in multiple applications and hosting environments;

[11] Figure 5 illustrates a computer-implemented method of generating a componentized user interface in accordance with an embodiment of the invention;

[12] Figures 6 illustrates a user interface being initialized in accordance with the present invention;

[13] Figure 7 illustrates a plug-in being added to a plug-in directory of a user interface in accordance with an embodiment of the invention;

[14] Figure 8 illustrates a plug-in being selected and the user interface response in accordance with an embodiment of the invention;

[15] Figure 9 illustrates the addition of a second plug-in to the plug-in directory in accordance with an embodiment of the invention;

[16] Figure 10 illustrates a plug-in being selected and the user interface response in accordance with an embodiment of the invention;

[17] Figure 11 illustrates another plug-in being selected and the user interface response in accordance with an embodiment of the invention;

[18] Figure 12 illustrates the activation of a plug-in by selection of the plug-in window and the user interface response in accordance with an embodiment of the invention;

[19] Figure 13 illustrates the selection of a toolbar button in accordance with an embodiment of the invention; and

[20] Figure 14 illustrates the sending of a message by the framework in accordance with an embodiment of the invention.

DETAILED DESCRIPTION

Exemplary Operating Environment

[21] Figure 1 is a functional block diagram of an example of a conventional general-purpose digital computing environment that can be used to implement a componentized user interface in accordance with various aspects of the present invention. In Figure 1, a computer 100 includes a processing unit 110, a system memory 120, and a system bus 130 that couples various system components including the system memory to the processing unit 110. The system bus 130 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory 120 includes read only memory (ROM) 140 and random access memory (RAM) 150.

[22] A basic input/output system 160 (BIOS), containing the basic routines that help to transfer information between elements within the computer 100, such as during start-up, is stored in the ROM 140. The computer 100 also includes a hard disk drive 170 for reading from and writing to a hard disk (not shown), a magnetic disk drive 180 for reading from or writing to a removable magnetic disk 190, and an optical disk drive 191

for reading from or writing to a removable optical disk 192 such as a CD ROM or other optical media. The hard disk drive 170, magnetic disk drive 180, and optical disk drive 191 are connected to the system bus 130 by a hard disk drive interface 192, a magnetic disk drive interface 193, and an optical disk drive interface 194, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 100. It will be appreciated by those skilled in the art that other types of computer readable media that can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the example operating environment.

[23] A number of program modules can be stored on the hard disk drive 170, magnetic disk 190, optical disk 192, ROM 140 or RAM 150, including an operating system 195, one or more application programs 196, other program modules 197, and program data 198. A user can enter commands and information into the computer 100 through input devices such as a keyboard 101 and pointing device 102. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit 110 through a serial port interface 106 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). Further still, these devices may be coupled directly to the system bus 130 via an appropriate interface (not shown). A monitor 107 or other type of display device is also connected to the system bus 130 via an interface, such as a video adapter 108. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

[24] The computer 100 can operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 109. The remote computer 109 can be a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 100, although only a memory storage device 111 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 112 and a wide area network (WAN) 113. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[25] When used in a LAN networking environment, the computer 100 is connected to the local network 112 through a network interface or adapter 114. When used in a WAN networking environment, the personal computer 100 typically includes a modem 115 or other means for establishing a communications over the wide area network 113, such as the Internet. The modem 115, which may be internal or external, is connected to the system bus 130 via the serial port interface 106. In a networked environment, program modules depicted relative to the personal computer 100, or portions thereof, may be stored in the remote memory storage device.

[26] It will be appreciated that the network connections shown are illustrative and other techniques for establishing a communications link between the computers can be used. The existence of any of various well-known protocols such as TCP/IP, Ethernet, FTP, HTTP, Bluetooth, IEEE 802.11x and the like is presumed, and the system can be operated in a client-server configuration to permit a user to retrieve web pages from a web-based server. Any of various conventional web browsers can be used to display and manipulate data on web pages.

Description of Illustrative Embodiments

[27] Figure 2 illustrates a system 200 that provides a componentized and extensible user interface in accordance with an embodiment of the invention. A user interface framework 202 may allow a user to interact with a computer application or hosting environment such as Windows® Testing Technologies Studio 204 (WTT Studio). In the embodiment of Figure 2, the user interface framework 202 is being utilized with a single computer application or hosting environment 204. Those skilled in the art will realize that the system of Figure 2 can be utilized with numerous applications or hosting environments and that Windows® Testing Technologies Studio is utilized for purposes of illustration.

[28] In the embodiment of Figure 2, the user interface framework 202 loads various interface plug-ins 206 to customize the user interface according to the specific requirements of the user. For example, the customization of the user interface may create menu items, toolbar buttons, and status bar panels according to the instructions of the executed user interface plug-ins 206. The user interface framework 202 manages the user interface plug-ins 206. Specifically, the user interface plug-ins 206 may comprise modules that are activated by a user selecting menu choices on a user interface framework menu.

[29] The user interface framework 202 may be an overall construct that controls the hierarchical organization of the user interface plug-ins 206. Communications channels may be provided by Microsoft® .NET common language runtime platform so that the user interface plug-ins 206 may talk to each other and the user interface framework 202. In addition, the user interface framework 202 provides a first set of interface elements that may be common to all of the user interface plug-ins 206. For example, the user interface framework 202 may contain code for the opening and saving of files. In this manner, each user interface plug-ins 206 may not be required to duplicate functionality

that may be common to numerous user interface plug-ins 206, as the functionality may be located in the user interface framework 202.

[30] The user interface framework 202 may be configured to discover the user interface plug-ins 206 upon initialization through the utilization of a user interface component loader (not shown). The user interface component loader discovers the user interface plug-ins 206 and loads the use interface plug-ins 206 into a plug-in directory. Also, if the user interface framework encounters an error when loading a user interface plug-in, the user interface plug-in may be flagged and the user notified of the error. In addition, the user interface framework 202 may not be aware of the existence of a particular user interface plug-in until initialization or runtime.

[31] The user interface plug-ins 206 are hosted by a computer application that provides a shell to host the user interface plug-ins 206. For example, Figure 3 shows a screen shot of one embodiment of a user interface shell 302. The user interface shell 302 may be a typical MDI window that allows for the displaying of a menu bar 304, a toolbar 306, and a status bar 308 in the user area of the user interface shell 302. A shown in Figure 3 on the status bar 308, a plug-in named Plugin2 is active and the status bar has been customized for Plugin2. In addition, the menu bar 304 shows that two different plug-ins named Plugin1 and Plugin2 may have been loaded into user interface shell 302. A user can activate a plug-in by clicking a particular menu item. For example, a user can activate Plugin2 of Figure 3 by selecting the "Event Test – WTT Studio.NET Prototype" window 312. As a user interacts with a plug-in, the plug-in may dynamically update the user interface shell 302 according to its state change.

[32] Each user interface plug-in of the present invention may be implemented with the use of two files. A first file may provide an interface between the framework and the plug-in. The first file may be in the form of an executable file such as a DLL file. The second file

may be written in a mark-up language such as XML. The second file may describe the attributes of the plug-in. The second file may also comprise menu elements that may include a toolbar, a status bar, and a menu bar. One skilled in the art will realize that other mark-up languages can be utilized to describe the attributes of the plug-ins such as a standard generalized markup language (SGML).

[33] Figure 4 illustrates a system providing a componentized user interface for use in numerous computer applications and hosting environments. In the embodiment of Figure 4, a shell adapter interface 402 provides an interface between the user interface framework 202 and the various application specific adapters 404, 406, and 410. To support the utilization of user interface plug-ins 206 in a plurality of computer applications and hosting environments, a layer of abstraction may be introduced that insulates the user interface plug-ins 206 from the applications or hosting environments. In this manner, user interface plug-ins 206 may be utilized with minimal, if any, coding changes to the plug-ins 206. For example, Figure 4 illustrates three applications or hosting environments namely WTT Studio 204, Product Studio 408, and Visual Studio.NET 412 (VS.Net). Each of these applications WTT Studio 204, Product Studio 408, and VS.Net 412 may want to utilize plug-ins 206 in their hosting environments. One skilled in the art will appreciate that other numerous computer applications or hosting environments may also be utilized to implement aspects of the invention.

[34] In order for plug-ins 206 to be hosted in the various computer applications or hosting environments, an adapter for each application and hosting environment may be utilized. For example, a WTT Standalone Adapter 404 may enable WTT Studio 204 to utilize plug-ins 206 in the WTT Studio hosting environment 204. The WTT standalone adapter 404 may map the functions or interface elements of the user interface plug-ins 206 to corresponding functions or interface elements in WTT Studio hosting environment 204. The mapping may enable the user interface plug-ins 206 to be utilized in various

computer applications and hosting environments. For example, a Product Studio adapter 406 may enable the plug-ins 206 to be utilized in Product Studio 408. Similarly, a Visual Studio.Net adapter 410 may enable the plug-ins 206 to be utilized in the Visual Studio hosting environment 412.

[35] Figure 5 illustrates a computer-implemented method of Figure 4 for the generation of a componentized user interface in accordance with an embodiment of the invention. In step 502, a user interface framework 202 is provided having a first set of interface elements. The interface elements of user interface framework 202 may be comprised of interface elements that are common to plug-ins 206 eliminating duplicate functionality in each of the plug-ins 206. The utilization of the user interface framework 202 for common interface elements may save plug-in development time and memory allocation.

[36] In step 504, a second set of interface elements may be provided by a first plug-in that may be linked to the user interface framework 202. Similarly, in step 506, a third set of interface elements may be provided by a second plug-in that may be linked to user interface framework 202. In step 508, a shell linked to the user interface framework 202 may be provided to host the first and second plug-ins. In Figure 4, the shell may comprise WTT Studio 204, Product Studio 408, or VS.NET 412. Finally, in step 510 an interface may be provided between the shell and the first plug-in and between the shell and the second plug-in. The interface may be used in order to utilize the second and third set of interface elements. The interface may take the form of an adapter such as the WTT Standalone Adapter 404, PS Adapter 406, and VS.Net Adapter 410 of Figure 4.

[37] Figures 6 through 14 illustrate a componentized user interface in accordance with the present invention. In Figure 6, a user interface shell 602 is displayed that includes a menu bar 604 and a toolbar 606. As shown in Figure 6, the user interface shell 602 does not indicate the existence of any plug-ins. In Figure 7, a plug-in named 'Publisher' may

be added to a Plugins directory 720. A list of the added plug-ins to the user interface shell 702 may be discovered through the use of drop down menu 722. As illustrated in Figure 7, the user interface shell 702 may be updated or customized upon the addition of the 'Publisher' plug-in to the Plugins directory 720.

[38] In Figure 8, the 'Publisher' plug-in 820 has been started and the user interface shell 802 has been updated. As shown in Figure 8, the menu bar 804 and the tool bar 806 may be customized based on the loading of the 'Publisher' plug-in 820. Also, a MDI window 812 entitled 'Publisher' may be opened and active. As further illustrated in Figure 8, the a New listener button 814 may be grayed out as the 'Publisher' plug-in 820 may receive notification from the user interface framework 202 that a listener plug-in is not installed.

[39] Figure 9 illustrates the addition of a second plug-in to a plug-in directory in accordance with an embodiment of the invention. In Figure 9, a plug-in named 'Listener' may be added to a Plugins directory 920 on menu bar 904. A list of the added plug-ins to the user interface shell 902 may be discovered through the use of drop down menu 922. As illustrated in Figure 9, the user interface shell 902 may be updated or customized based upon the addition of the 'Listener' plug-in to the Plugins directory 920.

[40] In Figure 10 the 'Publisher' plug-in 820 may be activated and the user interface shell 1002 updated. As shown in Figure 10, the menu bar 1004 and the tool bar 1006 may be updated and customized. In addition, a MDI window 1012 may indicate that the 'Listener' plug-in has been loaded as the New listener button 1014 is active. If the New listener button 1014 is selected a listener may be started as illustrated in Figure 11. Figure 11 illustrates that a new listener may have been started as indicated by MDI window 1116, and the placing of the publisher MDI window 1118 in the background. In addition, the menu bar 1104 may be updated to indicate that the Listener plug-in 1120 is

active. Finally, a status bar 1122 on the user interface shell 1102 may also indicate that the Listener plug-in 1120 is active.

[41] Figure 12 illustrates that a plug-in may be activated by selecting the plug-ins MDI window. For example, in Figure 12 the Publisher MDI window 1218 may be selected from the user interface shell 1202. When the Publisher MDI window 1218 is selected, the menu bar 1204 and tool bar 1206 are updated. In addition, the user interface framework may notify the Publisher plug-in 820 of the existence of a listener instance 1220.

[42] In Figure 13, a listener instance 1320 may be selected in order to send a message. With the selection of the listener instance 1320 a send message tool bar button may be activated. In Figure 14, a message may be delivered 1422 to a listener. The MDI window 1422 may illustrate the message.

[43] The present invention has been described in terms of preferred and exemplary embodiments thereof. Numerous other embodiments, modifications and variations within the scope and spirit of the appended claims will occur to persons of ordinary skill in the art from a review of this disclosure.

[44] Attached is an exemplary schema for the present invention:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by Huaming Wang (Microsoft Corp) -->
<!--W3C Schema generated by XMLSPY v5 rel. 3 U (http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <!-- simple types -->
    <xs:simpleType name="dockstyle">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="fill"/>
            <xs:enumeration value="top"/>
            <xs:enumeration value="bottom"/>
            <xs:enumeration value="left"/>
            <xs:enumeration value="right"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="formtype">
        <xs:restriction base="xs:NMTOKEN">
```

```
            <xs:enumeration value="mdiparent"/>
            <xs:enumeration value="mdichild"/>
            <!-- xs:enumeration value="form"/>
            <xs:enumeration value="docking"/>
            <xs:enumeration value="panel"/>
            <xs:enumeration value="tab"/>
            <xs:enumeration value="tabpage"/-->
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="windowstate">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="maximized"/>
            <xs:enumeration value="minimized"/>
            <xs:enumeration value="normal"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="formborderstyle">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="fixeddialog"/>
            <xs:enumeration value="fixed3d"/>
            <xs:enumeration value="fixedsingle"/>
            <xs:enumeration value="fixedtoolwindow"/>
            <xs:enumeration value="none"/>
            <xs:enumeration value="sizable"/>
            <xs:enumeration value="sizabletoolwindow"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="shortcut">
        <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="Shortcut.Alt0"/>
            <xs:enumeration value="Shortcut.Alt1"/>
            <xs:enumeration value="Shortcut.Alt2"/>
            <xs:enumeration value="Shortcut.Alt3"/>
            <xs:enumeration value="Shortcut.Alt4"/>
            <xs:enumeration value="Shortcut.Alt5"/>
            <xs:enumeration value="Shortcut.Alt6"/>
            <xs:enumeration value="Shortcut.Alt7"/>
            <xs:enumeration value="Shortcut.Alt8"/>
            <xs:enumeration value="Shortcut.Alt9"/>
            <xs:enumeration value="Shortcut.AltBksp"/>
            <xs:enumeration value="Shortcut.AltF1"/>
            <xs:enumeration value="Shortcut.AltF2"/>
            <xs:enumeration value="Shortcut.AltF3"/>
            <xs:enumeration value="Shortcut.AltF4"/>
            <xs:enumeration value="Shortcut.AltF5"/>
            <xs:enumeration value="Shortcut.AltF6"/>
            <xs:enumeration value="Shortcut.AltF7"/>
            <xs:enumeration value="Shortcut.AltF8"/>
            <xs:enumeration value="Shortcut.AltF9"/>
            <xs:enumeration value="Shortcut.AltF10"/>
            <xs:enumeration value="Shortcut.AltF11"/>
            <xs:enumeration value="Shortcut.AltF12"/>
            <xs:enumeration value="Shortcut.Ctrl0"/>
            <xs:enumeration value="Shortcut.Ctrl1"/>
            <xs:enumeration value="Shortcut.Ctrl2"/>
            <xs:enumeration value="Shortcut.Ctrl3"/>
            <xs:enumeration value="Shortcut.Ctrl4"/>
            <xs:enumeration value="Shortcut.Ctrl5"/>
            <xs:enumeration value="Shortcut.Ctrl6"/>
            <xs:enumeration value="Shortcut.Ctrl7"/>
            <xs:enumeration value="Shortcut.Ctrl8"/>
```

```
<xs:enumeration value="Shortcut.Ctrl9"/>
<xs:enumeration value="Shortcut.CtrlF1"/>
<xs:enumeration value="Shortcut.CtrlF2"/>
<xs:enumeration value="Shortcut.CtrlF3"/>
<xs:enumeration value="Shortcut.CtrlF4"/>
<xs:enumeration value="Shortcut.CtrlF5"/>
<xs:enumeration value="Shortcut.CtrlF6"/>
<xs:enumeration value="Shortcut.CtrlF7"/>
<xs:enumeration value="Shortcut.CtrlF8"/>
<xs:enumeration value="Shortcut.CtrlF9"/>
<xs:enumeration value="Shortcut.CtrlF10"/>
<xs:enumeration value="Shortcut.CtrlF11"/>
<xs:enumeration value="Shortcut.CtrlF12"/>
<xs:enumeration value="Shortcut.CtrlDel"/>
<xs:enumeration value="Shortcut.CtrlA"/>
<xs:enumeration value="Shortcut.CtrlB"/>
<xs:enumeration value="Shortcut.CtrlC"/>
<xs:enumeration value="Shortcut.CtrlD"/>
<xs:enumeration value="Shortcut.CtrlE"/>
<xs:enumeration value="Shortcut.CtrlF"/>
<xs:enumeration value="Shortcut.CtrlG"/>
<xs:enumeration value="Shortcut.CtrlH"/>
<xs:enumeration value="Shortcut.CtrlI"/>
<xs:enumeration value="Shortcut.CtrlIns"/>
<xs:enumeration value="Shortcut.CtrlJ"/>
<xs:enumeration value="Shortcut.CtrlK"/>
<xs:enumeration value="Shortcut.CtrlL"/>
<xs:enumeration value="Shortcut.CtrlM"/>
<xs:enumeration value="Shortcut.CtrlN"/>
<xs:enumeration value="Shortcut.CtrlO"/>
<xs:enumeration value="Shortcut.CtrlP"/>
<xs:enumeration value="Shortcut.CtrlQ"/>
<xs:enumeration value="Shortcut.CtrlR"/>
<xs:enumeration value="Shortcut.CtrlS"/>
<xs:enumeration value="Shortcut.CtrlT"/>
<xs:enumeration value="Shortcut.CtrlU"/>
<xs:enumeration value="Shortcut.CtrlV"/>
<xs:enumeration value="Shortcut.CtrlW"/>
<xs:enumeration value="Shortcut.CtrlX"/>
<xs:enumeration value="Shortcut.CtrlY"/>
<xs:enumeration value="Shortcut.CtrlZ"/>
<xs:enumeration value="Shortcut.CtrlShift0"/>
<xs:enumeration value="Shortcut.CtrlShift1"/>
<xs:enumeration value="Shortcut.CtrlShift2"/>
<xs:enumeration value="Shortcut.CtrlShift3"/>
<xs:enumeration value="Shortcut.CtrlShift4"/>
<xs:enumeration value="Shortcut.CtrlShift5"/>
<xs:enumeration value="Shortcut.CtrlShift6"/>
<xs:enumeration value="Shortcut.CtrlShift7"/>
<xs:enumeration value="Shortcut.CtrlShift8"/>
<xs:enumeration value="Shortcut.CtrlShift9"/>
<xs:enumeration value="Shortcut.CtrlShiftF1"/>
<xs:enumeration value="Shortcut.CtrlShiftF2"/>
<xs:enumeration value="Shortcut.CtrlShiftF3"/>
<xs:enumeration value="Shortcut.CtrlShiftF4"/>
<xs:enumeration value="Shortcut.CtrlShiftF5"/>
<xs:enumeration value="Shortcut.CtrlShiftF6"/>
<xs:enumeration value="Shortcut.CtrlShiftF7"/>
<xs:enumeration value="Shortcut.CtrlShiftF8"/>
<xs:enumeration value="Shortcut.CtrlShiftF9"/>
<xs:enumeration value="Shortcut.CtrlShiftF10"/>
```

```
            <xs:enumeration value="Shortcut.CtrlShiftF11"/>
            <xs:enumeration value="Shortcut.CtrlShiftF12"/>
            <xs:enumeration value="Shortcut.CtrlShiftA"/>
            <xs:enumeration value="Shortcut.CtrlShiftB"/>
            <xs:enumeration value="Shortcut.CtrlShiftC"/>
            <xs:enumeration value="Shortcut.CtrlShiftD"/>
            <xs:enumeration value="Shortcut.CtrlShiftE"/>
            <xs:enumeration value="Shortcut.CtrlShiftF"/>
            <xs:enumeration value="Shortcut.CtrlShiftG"/>
            <xs:enumeration value="Shortcut.CtrlShiftH"/>
            <xs:enumeration value="Shortcut.CtrlShiftI"/>
            <xs:enumeration value="Shortcut.CtrlShiftJ"/>
            <xs:enumeration value="Shortcut.CtrlShiftK"/>
            <xs:enumeration value="Shortcut.CtrlShiftL"/>
            <xs:enumeration value="Shortcut.CtrlShiftM"/>
            <xs:enumeration value="Shortcut.CtrlShiftN"/>
            <xs:enumeration value="Shortcut.CtrlShiftO"/>
            <xs:enumeration value="Shortcut.CtrlShiftP"/>
            <xs:enumeration value="Shortcut.CtrlShiftQ"/>
            <xs:enumeration value="Shortcut.CtrlShiftR"/>
            <xs:enumeration value="Shortcut.CtrlShiftS"/>
            <xs:enumeration value="Shortcut.CtrlShiftT"/>
            <xs:enumeration value="Shortcut.CtrlShiftU"/>
            <xs:enumeration value="Shortcut.CtrlShiftV"/>
            <xs:enumeration value="Shortcut.CtrlShiftW"/>
            <xs:enumeration value="Shortcut.CtrlShiftX"/>
            <xs:enumeration value="Shortcut.CtrlShiftY"/>
            <xs:enumeration value="Shortcut.CtrlShiftZ"/>
            <xs:enumeration value="Shortcut.Del"/>
            <xs:enumeration value="Shortcut.F1"/>
            <xs:enumeration value="Shortcut.F2"/>
            <xs:enumeration value="Shortcut.F3"/>
            <xs:enumeration value="Shortcut.F4"/>
            <xs:enumeration value="Shortcut.F5"/>
            <xs:enumeration value="Shortcut.F6"/>
            <xs:enumeration value="Shortcut.F7"/>
            <xs:enumeration value="Shortcut.F8"/>
            <xs:enumeration value="Shortcut.F9"/>
            <xs:enumeration value="Shortcut.Ins"/>
            <xs:enumeration value="Shortcut.ShiftDel"/>
            <xs:enumeration value="Shortcut.ShiftF1"/>
            <xs:enumeration value="Shortcut.ShiftF2"/>
            <xs:enumeration value="Shortcut.ShiftF3"/>
            <xs:enumeration value="Shortcut.ShiftF4"/>
            <xs:enumeration value="Shortcut.ShiftF5"/>
            <xs:enumeration value="Shortcut.ShiftF6"/>
            <xs:enumeration value="Shortcut.ShiftF7"/>
            <xs:enumeration value="Shortcut.ShiftF8"/>
            <xs:enumeration value="Shortcut.ShiftF9"/>
            <xs:enumeration value="Shortcut.ShiftF10"/>
            <xs:enumeration value="Shortcut.ShiftF11"/>
            <xs:enumeration value="Shortcut.ShiftF12"/>
            <xs:enumeration value="Shortcut.ShiftIns"/>
        </xs:restriction>
    </xs:simpleType>
    <!-- emlement definitions -->
    <xs:element name="Assemblies">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Assembly" maxOccurs="unbounded"/>
            </xs:sequence>
```

```
    </xs:complexType>
</xs:element>
<xs:element name="Assembly">
    <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="ClientView">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Container"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="Set">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Assemblies"/>
            <xs:element ref="MenuItems"/>
            <xs:element ref="Group" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Disabled" type="xs:boolean" use="optional" default="false"/>
    </xs:complexType>
    <xs:key name="GroupKey">
        <xs:selector xpath="Group"/>
        <xs:field xpath="@Name"/>
    </xs:key>
    <xs:unique name="GroupUniqueness">
        <xs:selector xpath="Group"/>
        <xs:field xpath="@Name"/>
    </xs:unique>
    <xs:keyref name="GroupReference" refer="GroupKey">
        <xs:selector xpath=".//MenuItem"/>
        <xs:field xpath="@Target"/>
    </xs:keyref>
</xs:element>
<xs:element name="Group">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="MenuItems" minOccurs="0"/>
            <xs:element ref="ToolBarButtons" minOccurs="0"/>
            <xs:element ref="ClientView"/>
            <xs:element ref="StatusBarPanels" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Disabled" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="Name" type="xs:string" use="optional"/>
        <xs:attribute name="MultiInstance" type="xs:boolean" use="optional" default="false"/>
    </xs:complexType>
</xs:element>
<xs:element name="MenuItems">
    <xs:complexType>
        <xs:sequence maxOccurs="unbounded">
            <xs:element ref="MenuItem" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:unique name="MenuNameUniqueness">
        <xs:selector xpath=".//*"/>
        <xs:field xpath="@Name"/>
    </xs:unique>
```

```
</xs:element>
<xs:element name="Extension">
    <xs:complexType>
        <xs:attribute name="Type" type="xs:string" use="required"/>
        <xs:attribute name="Name" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="StatusBarPanels">
    <xs:complexType>
        <xs:sequence maxOccurs="unbounded">
            <xs:element ref="StatusBarPanel" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="ToolBarButtons">
    <xs:complexType>
        <xs:sequence maxOccurs="unbounded">
            <xs:element ref="ToolBarButton" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="UIClassicTabControl">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Extension" maxOccurs="unbounded"/>
            <xs:element ref="UIDocument" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string" use="optional"/>
        <xs:attribute name="Dock" type="dockstyle" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Container">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="UIWizard" minOccurs="0"/>
            <xs:element ref="UIClassicTabControl" minOccurs="0"/>
            <xs:element name="MainMenu" minOccurs="0">
                <xs:complexType>
                    <xs:attribute name="Name" type="xs:string" use="required"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="ToolBar" minOccurs="0">
                <xs:complexType>
                    <xs:attribute name="Name" type="xs:string" use="required"/>
                    <xs:attribute name="Dock" type="dockstyle" use="optional"/>
                    <xs:attribute name="Appearance">
                        <xs:simpleType>
                            <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="flat"/>
                                <xs:enumeration value="normal"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
            <xs:element name="StatusBar" minOccurs="0">
                <xs:complexType>
                    <xs:attribute name="Name" type="xs:string" use="required"/>
```

```
                    <xs:attribute name="Dock" type="dockstyle" use="optional"/>
                </xs:complexType>
            </xs:element>
        </xs:choice>
        <xs:attribute name="Name" type="xs:string" use="optional"/>
        <xs:attribute name="Type" type="formtype" use="required"/>
        <xs:attribute name="DefaultWidth" type="xs:positiveInteger" use="required"/>
        <xs:attribute name="DefaultHeight" type="xs:positiveInteger" use="required"/>
        <xs:attribute name="MaximizeBox" type="xs:boolean"/>
        <xs:attribute name="MinimizeBox" type="xs:boolean"/>
        <xs:attribute name="DisplayHeader" type="xs:string"/>
        <xs:attribute name="Modal" type="xs:boolean"/>
        <xs:attribute name="WindowState" type="windowstate"/>
        <xs:attribute name="FormBorderStyle" type="formborderstyle"/>
        <xs:attribute name="Dock" type="dockstyle"/>
    </xs:complexType>
</xs:element>
<xs:element name="MenuItem">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="MenuItem" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Target" type="xs:string" use="optional"/>
        <xs:attribute name="Name" type="xs:string"/>
        <xs:attribute name="DisplayName" type="xs:string" use="required"/>
        <xs:attribute name="Enabled" type="xs:boolean"/>
        <xs:attribute name="Shortcut" type="shortcut"/>
        <xs:attribute name="Icon" type="xs:string"/>
        <xs:attribute name="Index" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="-1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="MdiList" type="xs:boolean" use="optional" default="false"/>
    </xs:complexType>
</xs:element>
<xs:element name="StatusBarPanel">
    <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Text" type="xs:string"/>
        <xs:attribute name="BorderStyle">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="sunken"/>
                    <xs:enumeration value="raised"/>
                    <xs:enumeration value="none"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="Width" type="xs:positiveInteger" use="required"/>
        <xs:attribute name="MinimumWidth" type="xs:positiveInteger" use="required"/>
        <xs:attribute name="AutoSize" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="none"/>
                    <xs:enumeration value="spring"/>
                </xs:restriction>
            </xs:simpleType>
```

```
            </xs:attribute>
        </xs:complexType>
    </xs:element>
    <xs:element name="ToolBarButton">
        <xs:complexType>
            <xs:attribute name="Name" type="xs:string" use="optional"/>
            <xs:attribute name="DisplayName" type="xs:string" use="required"/>
            <xs:attribute name="Style" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:NMTOKEN">
                        <xs:enumeration value="pushbutton"/>
                        <xs:enumeration value="dropdownbutton"/>
                        <xs:enumeration value="separator"/>
                        <xs:enumeration value="togglebutton"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="Icon" type="xs:string"/>
            <xs:attribute name="Enabled" type="xs:boolean" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="UIWizard">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Extension" maxOccurs="unbounded"/>
                <xs:element ref="UIDocument" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="Name" type="xs:string" use="optional"/>
            <xs:attribute name="Dock" type="dockstyle" use="optional"/>
            <xs:attribute name="Icon" type="xs:string"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="UIDocument">
        <xs:complexType>
            <xs:attribute name="Name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

[45] Attached is an exemplary schema of a configuration file for plug-in of the present invention:

```
<?xml version="1.0" encoding="UTF-8"?>
<Set Name="WTTUI.Jobs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <Assemblies>
        <Assembly Name="Jobs"/>
        <Assembly Name="Jobx"/>
    </Assemblies>
    <MenuItems>
        <MenuItem Name="WTTJobs" DisplayName="Jobs" Enabled="true" Shortcut="Shortcut.F3" Icon="Jobs.ico">
            <MenuItem DisplayName="Explore" Target="Explore"/>
            <MenuItem DisplayName="-"/>
            <MenuItem DisplayName="Schedule">
                <MenuItem DisplayName="Schedule Wizard" Target="ScheduleWizard"/>
                <MenuItem DisplayName="Schedule Classic" Target="ScheduleClassicView"/>
            </MenuItem>
        </MenuItem>
    </MenuItems>
```

```
    <Group Name="Explore" Disabled="true">
        <ToolBarButtons>
            <ToolBarButton Name="Delete" DisplayName="Delete" Style="pushbutton" Icon="delete.ico" Enabled="false"/>
            <ToolBarButton Name="Rename" DisplayName="Rename" Style="pushbutton" Icon="rename.ico" Enabled="false"/>
        </ToolBarButtons>
        <ClientView>
            <Container Type="mdichild" DefaultWidth="808" DefaultHeight="592" MaximizeBox="true" MinimizeBox="true"
DisplayHeader="Job Explorer - WTT Studio.NET" Modal="false" WindowState="maximized" FormBorderStyle="fixed3d" Dock="fill">
                <Extension Type="UIJobTreeControl" Name="UIJobTreeControl"></Extension>
                <Extension Type="UIJobListControlWithoutQB" Name="UIJobListControl"></Extension>
            </Container>
        </ClientView>
        <StatusBarPanels>
            <StatusBarPanel Name="Database" Text="" BorderStyle="sunken" Width="100" MinimumWidth="10"
AutoSize="none"/>
            <StatusBarPanel Name="QueryStatus" Text="" BorderStyle="sunken" Width="300" MinimumWidth="30"
AutoSize="spring"/>
        </StatusBarPanels>
    </Group>
    <Group Name="ScheduleWizard">
        <ClientView>
            <Container Type="mdichild" Dock="top" DefaultWidth="682" DefaultHeight="490" FormBorderStyle="fixeddialog"
MinimizeBox="false" MaximizeBox="false" DisplayHeader="Schedule Wizard - WTT Studio.NET">
                <UIWizard Dock="fill" Icon="wtt.ico">
                    <Extension Type="UIScheduleWizardWelcome" Name="UIScheduleWizardWelcome"></Extension>
                    <Extension Type="UIScheduleWizardRolesControl" Name="UIScheduleWizardRolesControl"></Extension>
                    <Extension Type="ScheduleDocument" Name="ScheduleDocument"></Extension>
                    <Extension Type="UITCMController" Name="UITCMController"></Extension>
                </UIWizard>
            </Container>
        </ClientView>
    </Group>
    <Group Name="ScheduleClassicView">
        <ClientView>
            <Container Type="mdichild" DefaultWidth="682" DefaultHeight="490" DisplayHeader="Job Edit Classic - WTT
Studio.NET">
                <UIClassicTabControl Name="UIClassicTabControl" Dock="fill">
                    <Extension Type="UIScheduleAddJobsControl" Name="Add Jobs"></Extension>
                    <Extension Type="UIScheduleWizardSelectLab" Name="Machines"></Extension>
                    <Extension Type="ScheduleDocument" Name="ScheduleDocument"></Extension>
                    <Extension Type="UITCMController" Name="UITCMController"></Extension>
                </UIClassicTabControl>
            </Container>
        </ClientView>
    </Group>
</Set>
```

[46] Attached is an exemplary schema for a configuration file for a user interface framework shell for the present invention:

```
<?xml version="1.0" encoding="utf-8"?>
<Set Name="WTTUI.Framework" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <Assemblies>
        <Assembly Name="WTTUIBase"/>
        <Assembly Name="Icons"/>
    </Assemblies>
    <MenuItems>
        <MenuItem DisplayName="File" Index="0">
```

```
            <MenuItem DisplayName="Add Plugin" Target="AddPluginWizard"/>
            <MenuItem DisplayName="Plugin Manager" Target="PluginManager"/>
            <MenuItem DisplayName="-"/>
            <MenuItem DisplayName="Exit"/>
        </MenuItem>
        <MenuItem DisplayName="Help" Index="-1">
            <MenuItem DisplayName="About" Target="About"/>
        </MenuItem>
    </MenuItems>
    <Group Name="Shell">
        <ToolBarButtons>
            <ToolBarButton Name="Trace" DisplayName="Trace" Style="pushbutton" Icon="Trace.ico"/>
        </ToolBarButtons>
        <ClientView>
            <Container Type="mdiparent" DefaultHeight="600" DefaultWidth="800" DisplayHeader="WTT Studio.NET">
                <MainMenu Name="MainMenu"/>
                <ToolBar Name="MainToolBar" Dock="top" Appearance="flat"/>
                <StatusBar Name="UIStatusBar" Dock="bottom"/>
                <Extension Type="UITraceOutputControl" Name="UITraceOutputControl"></Extension>
            </Container>
        </ClientView>
    </Group>
    <Group Name="About">
        <ClientView>
            <Container Type="mdichild" DefaultWidth="540" DefaultHeight="420" FormBorderStyle="fixeddialog"
MinimizeBox="false" MaximizeBox="false" DisplayHeader="About WTT Studio.NET">
                <Extension Type="UIAboutControl" Name="About"></Extension>
            </Container>
        </ClientView>
    </Group>
    <Group Name="AddPluginWizard">
        <ClientView>
            <Container Type="mdichild" DefaultWidth="682" DefaultHeight="490" FormBorderStyle="fixeddialog"
MinimizeBox="false" MaximizeBox="false" DisplayHeader="Add Control Wizard - WTT Studio.NET">
                <UIWizard Name="UIWizard" Dock="fill" Icon="wtt.ico">
                    <Extension Type="UIAddControlWizardWelcome" Name="UIAddControlWizardWelcome"></Extension>
                    <Extension Type="UIAddControlWizardLoadXml" Name="UIAddControlWizardLoadXml"></Extension>
                    <Extension Type="UIAddControlWizardSummary" Name="UIAddControlWizardSummary"></Extension>
                </UIWizard>
            </Container>
        </ClientView>
    </Group>
    <Group Name="PluginManager">
        <ClientView>
            <Container Type="mdichild" DefaultWidth="682" DefaultHeight="490" FormBorderStyle="fixeddialog"
MinimizeBox="false" MaximizeBox="false" DisplayHeader="User Control Manager - WTT Studio.NET">
                <Extension Type="UIControlManager" Name="ControlManager"></Extension>
            </Container>
        </ClientView>
    </Group>
</Set>
```